The software behind the Higgs boson discovery

David Rousseau[1,*]

[1] *Laboratoire de l'Accélérateur Linéaire, Université Paris-Sud and CNRS/IN2P3, Orsay, France*

This article describes the enormous software development effort associated with teasing out evidence for the elusive Higgs boson, a cornerstone of the Standard Model.

Published in IEEE Software, vol. 29, no. 5, pp. 11-15, Sept.-Oct. 2012
DOI 10.1109/MS.2012.123
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6276293

* rousseau@lal.in2p3.fr http://www.linkedin.com/pub/david-rousseau/56/ba7/b47

The Atlas experiment [1], together with the Compact Muon Solenoid experiment [2], recently claimed the discovery of a new particle that is very likely the Higgs boson. This particle was theorized almost 50 years ago to have the very special role to give mass to other elementary particles. (This is the final ingredient of the Standard Model of particle physics, ruling subatomic particles and forces.) The experiment sits on the Large Hadron Collider (LHC) at CERN (the European Organization for Nuclear Research), Geneva, which began operating in 2009, after about 20 years of design and construction, and will continue operating for at least the next 10 years.

This column describes what Atlas's software does, how it is developed and tested, and the very significant computational and data management challenges it presents.

FROM COLLIDER TO LAPTOP

When operating, the experiment outputs a constant flux of 0.5 Gbytes per second of raw data, which must be digested, reduced, and distributed to 3,000 physicists in 170 laboratories worldwide for analysis.

Collision

Every 50 nanoseconds, the LHC collides bunches of protons within each of its experiments. Two protons colliding produce a small explosion, where part of the protons kinematical energy is converted into new particles. Most of the resulting particles are very unstable and decay quickly into a cascade of lighter particles, which the Atlas detector measures. Physicists designed the detector to identify and measure the particles energy and direction, which allows them to climb back up the cascade of particles to reach evidence for the primary, heaviest ones. For example, the Higgs boson might decay into two Z bosons, which themselves can decay into four electrons or muons (an unstable particle living long enough to sail through the whole detector).

For each bunch of collisions, a complex three-stage trigger system reduces the rate from 20 million collisions to about 500 per second by looking for signatures of high-energy particles standing out from the crowd of less interesting, lower-energy ones. For each selected collision, a raw event of about 1 Mbyte is written on disk, listing the scattered energy deposit in the different layers of the detector. More than 1 billion such events are recorded each year, which equates to about a petabyte of data.

RECONSTRUCTION

After all the different detectors measure a particular collision, the system reconstructs a collection of about 100 particles as accurately as possible from the raw information, with their type, energy, and direction (see Figure 1). In practice, the reconstruction stage consists of more than 200 different algorithms and takes

---

[1] http://atlas.ch
[2] http://cms.web.cern.ch

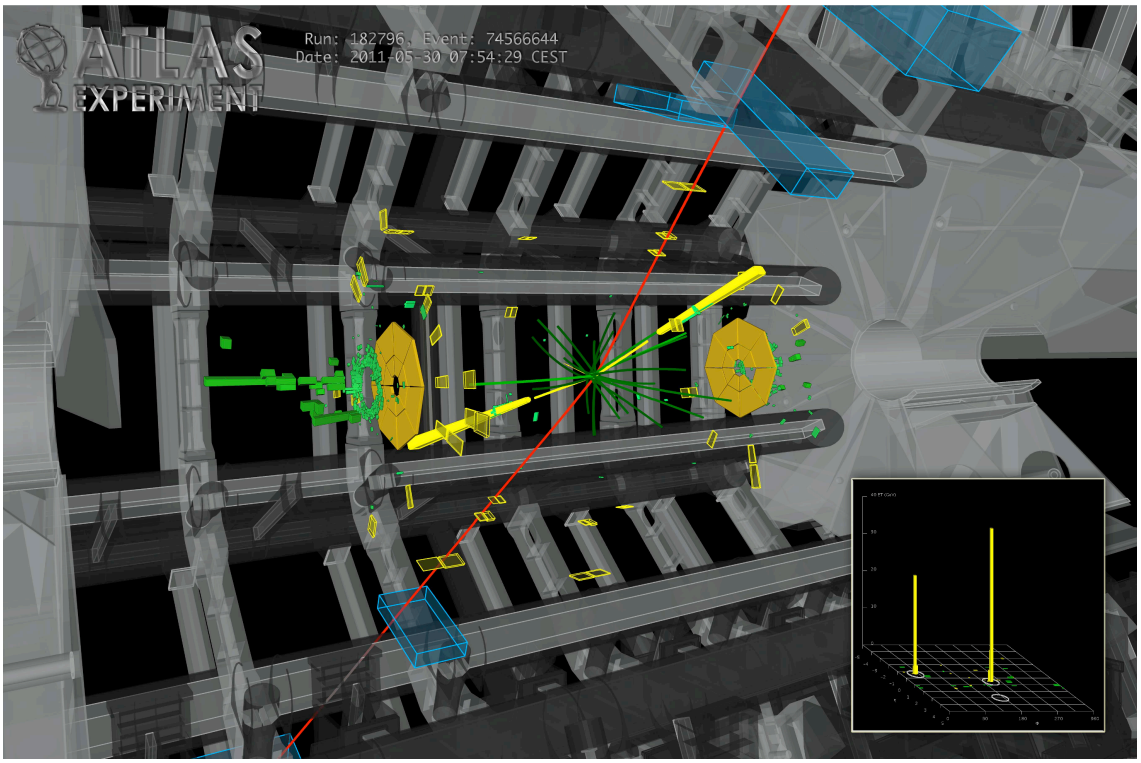

FIG. 1: An event display of a Higgs boson candidate. The red and yellow tracks are consistent with being decay products of two Z bosons resulting from a Higgs boson decay.[a]

[a] The ATLAS collaboration, ATLAS-CONF-2011-162 https://cdsweb.cern.ch/record/1406357

about 20 seconds on a regular CPU core with 2 Gbytes of memory. A farm of up to 6,000 standard CPU cores at the CERN computing center processes the data streaming out of the detector within days.

Meanwhile, the reconstructed events are distributed worldwide, and thousands of histograms are filled with well-known quantities and monitored semiautomatically for a significant departure that would indicate a problem with the data. The process lets physicists analyze an event with confidence in less than one week after it's recorded.

Simulation

Everything we know about particle physics (as well as alternate theories) has been encoded in generators developed by specialists. In a probabilistic way, these generators reproduce the small explosions from the proton collisions. The resulting particles are then tracked through a virtual model of the detector. This process yields simulated events very close to real ones, which are then fed into the standard reconstruction stage. This simulation stage has been necessary in the past to prepare the reconstruction algorithms and the physics analyses before data acquisition begins. We still use it to test analysis ideas and to evaluate correction factors with many data simulation agreement cross checks. In particular, a large effort went into considering, both at the simulation and reconstruction stages, the features of a real detector rather than an

idealized one. For example, the detector is built with positional inaccuracies off by a fraction of millimeters, whereas trajectories need to be reconstructed with a precision within 10 microns. Also, a small percentage of the electronic channels misbehave (no signal, inaccurate signal, noisy signal, and so on) and need to be dealt with almost on a case-by-case basis. These effects are corrected using a wealth of self-calibrating algorithms, for example, by measuring systematic deviation of energy measurement of the same particle in one part of the detector with respect to another. Despite continual trade-off between accuracy and CPU time, the simulation is very CPU intensive, typically 1,000 seconds per event, mainly because it must handle up to 10 billion intermediate particles at a time. Until now, the number of simulated events generated was the same as the number of real events recorded.

Analysis

Analysis files result from the constant negotiation between different analysis groups and the central management, which arbitrates between needs and resource limitations. About a petabyte of real and simulated data analysis files is the input for all analyses done in the collaboration (the Higgs boson being only one piece of the overall experimental program).

Each group then defines derived analysis datasets over which it has full control, provided its volume fits its allocated resource. A key element is flexibility: by definition, the analysis activity is fundamental research, where creativity is encouraged; small discoveries are made every day, and new ideas should be tested quickly. Going through the petabyte of the full analysis dataset typically takes one week in real time, so the compromise between flexibility and speed is to define intermediate derived datasets. For example, the first one preselects events and variables and computes derived information to obtain a derived dataset of one terabyte, which can be analyzed overnight. The other one would ideally fit on the physicists laptop to allow for data mining in a few seconds, similar to an Internet search query. This is how the first histogram bumps were seen, later becoming a major discovery.

THE ATLAS FRAMEWORK

Except for the initial reconstruction stage, Atlas different processing stages (including simulation) are performed on the Atlas grid sites. About 100 computing centers worldwide have been certified as part of the trusted Atlas grid centers. Each site offers CPU and disk resources and has the latest Atlas software preinstalled. Copies of the analysis datasets are spread sparsely throughout these centers. Physicists submit tasks to the grid by specifying the input dataset to be processed and the exact configuration of the software to be run (which often includes their own developments). The grid system then automatically splits the task into separate jobs (typically, one job is a single-core process consuming or producing at most a few Gbytes of data and lasting at most a few hours) and sends the job to where the dataset sits. In recent months, more than 100,000 cores have been constantly running for Atlas processing.

The framework of all the Atlas software follows a whiteboard architecture implemented in C++. One event is loaded in memory, and then algorithms are run sequentially, reading one or more object collections from the whiteboard and writing one or more output collections. After all the algorithms have run, the final collections are written to disk and the next event is loaded. This is a single-thread process running on

standard processors under Linux. The C++ design for the transient whiteboard objects is relatively simple and stable: the objects have a well-defined physical meaning, and the complexity of the processing resides in the algorithms. Figure 2 illustrates the structure of the data flow of the reconstruction. The configuration is done through a Python layer for flexible configuration, so one traditional algorithm can be unplugged and replaced by a more modern, boosted decision tree algorithm with a one-line switch.

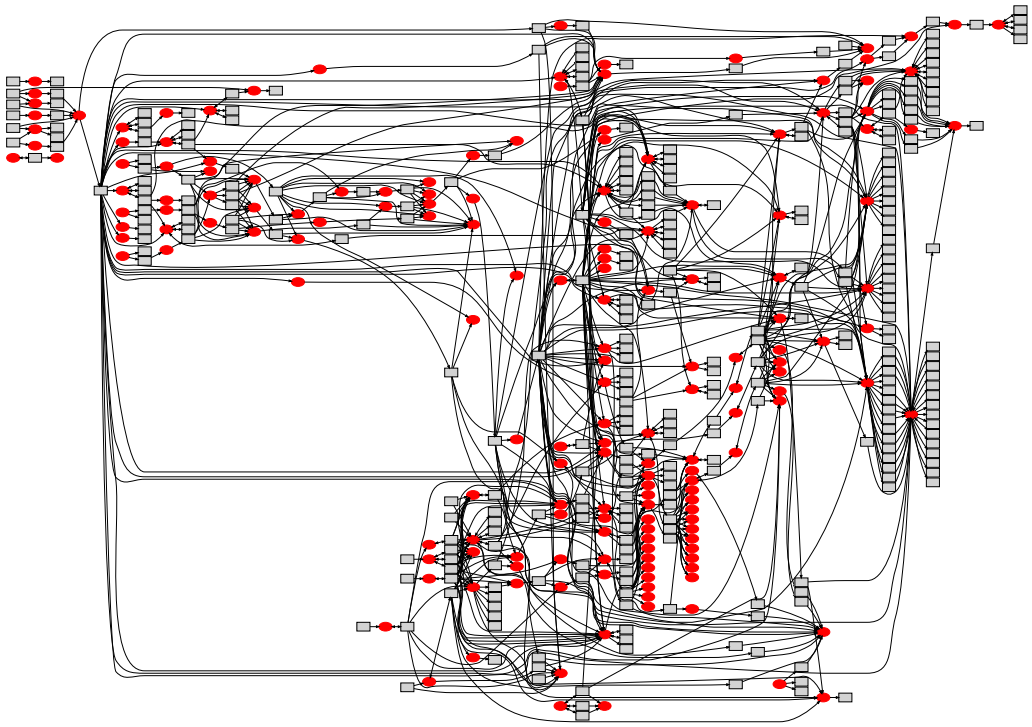

FIG. 2: The data flow of the reconstruction stage from left to right. Red ellipses represent algorithms and grey rectangles represent whiteboard objects..

CONTINUOUS SOFTWARE DEVELOPMENT

The software has about 4 million LOC, split among 2,000 packages. Approximately 1,500 users are regularly running at least some part of the code. It has been written by more than 1,000 developers (often the same people as the users), 250 being active in the last six months and submitting an average of 25 package changes per day. Its constantly evolving, with improvements, cleanup, and bug fixing. The developers have varying programming skills, from arch experts to PhD students fresh from C++ 101 courses. Furthermore, most are physicists more interested in tuning their algorithms than by software quality or resource usage.

Although formal code reviews have been organized in the past, they were consuming too much time for the few real software experts available. All of our software is open source (within the experiment). Newcomers are mentored by experts in the domain, who indicate where to insert contributions and which classes to use

as models.

As soon as they commit their code, it can be scrutinized by anyone in the experiment. Additionally, we rely on several layers of a posteriori control. We encourage people to submit new versions of their packages as often as possible, provided they are at least as functional as previous versions. The candidate future release is built every night with the new contributions. Compile failure as well as failure to satisfy some software quality metrics will result in automatic email notifications to developers.

Integration tests that run the most frequent configurations also run automatically. However, there's no way to automatically track failure in these tests to faulty packages, so people must take turns analyzing these failures and submitting bug reports, from which the faulty package change can be identified, rejected, and later fixed. Overall, developers are serious about testing their software before submitting it, but the test system is invaluable in spotting unforeseen side effects.

For patch releases, we have additional automatic regression tests checking bit by bit that the output is identical to a reference release. This system safely enables patch release improvements to reduce resource usage, as well as code cleanup, memory-leak removal, or fixes for very rare crashes.

Meanwhile, the next major Atlas release is being prepared, which includes significantly improved reconstruction algorithms, more accurate simulation to fix discrepancies observed with real data, and more accurate calibration constants. We have one such major release each year, which is immediately followed by a general reprocessing of all data events and regeneration of all simulated events. To validate this release, a team of people with expertise covering all aspects of the applications both process and scrutinize a set of reference samples. We also seek feedback from the working groups. The new release is iteratively patched until all experts sign it off.

OUTLOOK

The LHC will be revamped in the next two years, restarting with almost double its current energy. It's our responsibility to improve the efficiency of the algorithms to reach the ultimate performance potential of our detector. We'll take this opportunity to clean up the obsolete code and revise the design of whiteboard objects, which has been frozen for several years for the sake of stability. Last but not least, we will embrace parallelism. Our basic computing unit is the processing of one event, which takes between 1 millisecond and 1,000 seconds. So far, we've been able to adapt easily to multicore processors simply by running one job per core in parallel on the same processor. To exploit future many-core processors, however, we must change this paradigm and parallelize our framework and algorithms without forgetting that most of our developers will want to remain focused on future discoveries.